Pseudorandom Generators against CFL/n

Tomoyuki Yamakami

School of Computer Science and Engineering, University of Aizu
90 Kami-Iawase, Tsuruga, Ikki-machi, Aizu-Wakamatsu
Fukushima 965-8580, Japan

Abstract. Pseudorandomness has played a central role in modern cryptography, finding theoretical and practical applications to various fields of computer science. A function that generates such pseudorandom strings from shorter but truly random seeds is known as a pseudorandom generator. Our generators are designed to fool languages, rather than probabilistic algorithms. In particular, our generators take context-free languages with advice as their adversaries. We present an explicit example of such a pseudorandom generator, which can be also computed by a single-tape deterministic Turing machine running in time $O(n^2)$. In contrast, we show that there is no almost 1-1 pseudorandom generator against even context-free languages (without advice) if we demand it should be computed by a nondeterministic pushdown automaton equipped with a write-only output tape. Our proofs are all elementary, requiring no complicated proof techniques as in a polynomial-time setting, and utilize a specific feature of nondeterministic pushdown automata, which is interesting on its own light.

Keywords: context-free language, gap pseudorandom language, pseudorandom generator, pushdown automaton, swapping property

ACM Subject Classification: F.4.3, F.1.1, F.1.3

1 Introduction

In early 1980s, Blum and Micali [3] proposed a generator that produces a sequence in which any reasonable adversary hardly predicts the sequence's next bit. Although fundamentally equivalent, Yao [10] considered a generator that produces a sequence which no adversary distinguishes from a uniformly random sequence with a small margin of error. Such a generator is known as a *pseudorandom generator*, which has played as an important cryptographic primitive. The existence of a (polynomial-time computable) pseudorandom generator is, unfortunately, unknown unless we impose certain unproven complexity-theoretical assumptions.

As a main theme of this paper, we study a specific type of pseudorandom generator, whose adversaries are represented in a form of "languages" (or their "characteristic functions"), compared to standard "probabilistic algorithms." Such a generator also appears when the generator's adversaries are "Boolean circuits." As our limited adversaries, we consider most fundamental languages in formal language theory—regular languages and context-free languages, which have been extensively studied since the 1950s. An immediate advantage of dealing with such weak adversaries is that we can obtain corresponding pseudorandom generators without any unproven assumption.

Intuitively, a function G, which stretches n-bit seeds to $s(n)$-bit long strings, is said to *fool* a language A over the binary alphabet $\Sigma = \{0, 1\}$ if the characteristic function* χ_A of A cannot distinguish between the output distribution $\{G(x)\}_{x\in\Sigma^n}$ of G and a truly random distribution $\{y\}_{y\in\Sigma^{s(n)}}$ with non-*negligible* success probability. We call G a *pseudorandom generator* against a language family $\mathcal{C}$ if G fools every language A over Σ in $\mathcal{C}$.

A natural question is whether there exists an "easy-to-compute" pseudorandom generator against low-complexity languages. It was proven in [9] that a certain function computed by *nondeterministic pushdown automata* (or npda's) equipped with write-only output tapes (the set of those functions is denoted $\mathrm{CFLSV_t}$, similar to $\mathrm{NPSV_t}$) can be a pseudorandom generator even against the advised class REG/n (which is, loosely speaking, the family of regular languages supplemented, in parallel to inputs, by advice strings of size n, when n is the size of input [7, 8]). Moreover, such a generator can be "almost" 1-1 with a stretch factor exactly $n + 1$. The existence of such a pseudorandom generator is an evidence that a complexity gap between CFL

*The characteristic function χ_A of a language A is defined as $\chi_A(x) = 1$ if $x \in A$ and $\chi_A(x) = 0$ otherwise, for every input string x.

and REG/n is considerably wide. Can we make such a generator much easier-to-compute? Unfortunately, no almost 1-1 pseudorandom generator against REG (regular language family) can be computed by a single-tape linear-time off-line Turing machine [9].

As a natural extension of REG, we turn our attention to CFL, the family of context-free languages. Our main question is what the computational complexity of pseudorandom generators against the advised class CFL/n is, where, similar to REG/n, CFL/n is obtained from CFL by providing advice in parallel to inputs. Notice that CFL/n is quite different from REG/n; for instance, CFL/$n \neq$ co-CFL/n [8] whereas REG/n = co-REG/n. A simple way to construct a pseudorandom generator of a desired type is to use a so-called *diagonalization technique*: first enumerate all advised languages in CFL/n and then diagonalize them one by one to determine an outcome of the generator. However, such a method provides us only with a generator of significantly high complexity.

In this paper, however, we shall give an explicit example of a pseudorandom generator in 1-FTIME($O(n^2)$) against CFL/n, where 1-FTIME($t(n)$) is the set of functions computable by single-tape one-head off-line Turing machines running within time $t(n)$. Our generator does not involve any diagonalization-type construction and our proof of the generator's pseudorandomness is elementary, requiring no complex arguments usually found in a polynomial-time setting. For our proof, we require only two previously known results: a discrepancy upper bound of the inner-product-modulo-two function and a behavioral property of npda's. In particular, from such a property, we can derive a so-called *swapping property* of npda's, which is also interesting on its own light. To counter this example, additionally, we shall prove that no almost 1-1 pseudorandom generator even against CFL can be computed by npda's with write-only output tapes.

2 Fundamental Notions and Notations

Let $\mathbb{N}$ denote the set of all *nonnegative integers*. A function from $\mathbb{N}$ to $\mathbb{R}^{\geq 0}$ (nonnegative reals) is *negligible* if, for every non-zero polynomial p, $\mu(n) \leq 1/p(n)$ for all but finitely many numbers n in $\mathbb{N}$. For any two sets A and B, their *symmetric difference* $A \triangle B$ is the set $(A - B) \cup (B - A)$.

Let Σ be our alphabet (i.e., a finite nonempty set). A string x is a finite sequence of symbols taken from Σ. The empty string is always denoted λ. The length of a string x, denoted $|x|$, is the number of symbols in x. Let Σ^* be the set of all strings over Σ. For each number $i \in \mathbb{N}$, the notation Σ^n (resp., $\Sigma^{\leq n}$) denotes the set of all strings of length exactly n (resp., less than or equal to n). A language over Σ is a subset of Σ^*. For a language S over Σ and any number $n \in \mathbb{N}$, $dense(S)(n)$ denotes the cardinality of the set $S \cap \Sigma^n$. The notation χ_A denotes the *characteristic function* of A; namely, $\chi_A(x) = 1$ if $x \in A$ and $\chi_A(x) = 0$ otherwise.

For any string x of length n, let $pref_i(x)$ denote the string consisting of the first i symbols of x and similarly let $suf_j(x)$ be the string made up from the last j symbols of x. Moreover, we denote by $midd_{i,j}(x)$ the string obtained from x by deleting the first i symbols and the last $n - j$ symbols.

Let REG and CFL denote respectively the family of regular languages and the family of context-free languages. It is well known that regular languages and context-free languages are characterized by *deterministic finite automata* (or dfa's) and *nondeterministic pushdown automata* (or npda's), respectively.

An *advice function* is a map f from $\mathbb{N}$ to Γ^*, where Γ is an appropriate alphabet. A language L over an alphabet Σ is in an advised class $\mathcal{C}/n$ if there exist another alphabet Γ, an advice function h from $\mathbb{N}$ to Γ^*, and a language $S \in \mathcal{C}$ over Γ such that, for every string $x \in \Sigma^*$, $x \in L$ iff $[\begin{smallmatrix} x \\ h(|x|) \end{smallmatrix}] \in S$, where $[\begin{smallmatrix} x \\ y \end{smallmatrix}]$ denote a string made from x and y in parallel [7]. More precisely, for any pair of symbols $\sigma \in \Sigma_1$ and $\tau \in \Sigma_2$, the notation $[\begin{smallmatrix} \sigma \\ \tau \end{smallmatrix}]$ denotes a new symbol made from σ and τ. For two strings $x = x_1 x_2 \cdots x_n$ and $y = y_1 y_2 \cdots y_n$ of the same length n, the notation $[\begin{smallmatrix} x \\ y \end{smallmatrix}]$ is shorthand for the string $[\begin{smallmatrix} x_1 \\ y_1 \end{smallmatrix}][\begin{smallmatrix} x_2 \\ y_2 \end{smallmatrix}] \cdots [\begin{smallmatrix} x_n \\ y_n \end{smallmatrix}]$.

We assume that the reader is familiar with fundamental definitions and properties of *nondeterministic pushdown automata* (or ndpa's). For our convenience, we always assume that an input tape has two *endmarkers*, which surround an input string. See Section 6 for more details. We also use a model of a *single-tape one-head off-line Turing machine*, which is used to accept/reject an input string or to produce an output on this single tape. Let 1-FTIME($t(n)$) denote the set of all single-valued total functions computable by those single-tape one-head off-line deterministic Turing machines running in time at most $t(n)$. In particular, we write 1-FLIN for 1-FTIME($O(n)$). Moreover, we introduce $\mathrm{CFLSV_t}$ as the set of all single-valued total functions computed by npda's that are equipped with single *write-only* output tapes whose heads cannot go back to read already written symbols, provided that a string written on a single output tape along an npda's computation path is *valid* if the path is an accepting computation path [9].

3 Pseudorandom Generators and the Main Result

We shall explicitly state our main result of this paper. We formally introduce the notion of *pseudorandom generator* whose adversaries are languages (or their associated characteristic functions): particularly, context-free languages with advice.

Let $\Sigma = \{0,1\}$. We say that a function G from Σ^* to Σ^* has a *stretch factor* $s(n)$ if $|G(x)| = s(|x|)$ holds for any string $x \in \Sigma^*$. We use the notation $\mathrm{Prob}_{x\in\Sigma^n}[\mathcal{P}(x)]$ to denote the probability, over a random variable x distributed uniformly over Σ^n, that the property $\mathcal{P}(x)$ holds. When the probability space Σ^n is clear from the context, we omit the script "Σ^n" altogether.

Definition 3.1 A function G is said to *fool* a language A over Σ if the function $\ell(n) = |\mathrm{Prob}_x[\chi_A(G(x)) = 1] - \mathrm{Prob}_y[\chi_A(y) = 1]|$ is negligible, where x and y are random variables over Σ^n and $\Sigma^{s(n)}$, respectively. A function G is called a *pseudorandom generator* against a language family $\mathcal{C}$ if G fools every language A over the alphabet Σ in $\mathcal{C}$.

In this paper, we are mostly focused on generators whose stretch factor is $n+1$. Such a generator G is called *almost 1-1* if there is a negligible function $\tau(n) \geq 0$ such that $|\{G(x) \mid x \in \Sigma^n\}| = |\Sigma^n|(1-\tau(n))$ for all numbers $n \in \mathbb{N}$.

The existence of pseudorandom generators against REG/n was briefly discussed in [9]; however, it has been unknown whether there exists an "easy-to-compute" pseudorandom generator against CFL/n. We begin with stating a positive result: the existence of such a generator in 1-FTIME($O(n^2)$).

Theorem 3.2 [main theorem] *There exists an almost 1-1 pseudorandom generator in* 1-FTIME($O(n^2)$) *against* CFL/n *with the stretch factor* $n+1$.

Theorem 3.2 can contrast with the following negative result, which indicates a computational limitation of pseudorandom generators even against the language family CFL (and thus against CFL/n).

Proposition 3.3 *There is no almost 1-1 pseudorandom generator with the stretch factor* $n+1$ *in* $\mathrm{CFLSV_t}$ *against* CFL.

The rest of this paper is devoted to prove Theorem 3.2 and Proposition 3.3. In Section 4, with help of a notion of gap pseudorandomness, we shall give the proof of Proposition 3.3. In Sections 5–8, we shall present the proof of Theorem 3.2.

4 Gap Pseudorandom Languages

Before giving the proofs of Theorem 3.2 and Proposition 3.3, we shall discuss so-called *gap pseudorandom languages* and their close connection to our pseudorandom generators. We shall use this connection to prove the desired results in later sections.

Meanwhile, we use the notation Σ to denote any alphabet with $|\Sigma| \geq 2$ and let $\mathcal{C}$ be any language family. In [9], the notion of *$\mathcal{C}$-pseudorandom language*[†] was introduced in connection to pseudorandom generators against $\mathcal{C}$. This connection, however, is based on the closure property of $\mathcal{C}$ under complementation. Because CFL/n is not closed under complementation [8], we cannot use this connection directly. Instead, we consider a slightly weaker notion of *gap $\mathcal{C}$-pseudorandomness*, which provides a similar connection without any closure property.

Definition 4.1 A language L over an alphabet Σ is called *gap $\mathcal{C}$-pseudorandom* if, for every language A over Σ in $\mathcal{C}$, the function $\ell''(n) = \frac{|dense(L\cap A)(n) - dense(\overline{L}\cap A)(n)|}{|\Sigma^n|}$ is negligible; that is, for any non-zero polynomial p, there exists a positive number n_0 such that $\ell''(n) \leq 1/p(n)$ for all numbers $n \geq n_0$. For any language family $\mathcal{D}$, we say that $\mathcal{D}$ is *gap $\mathcal{C}$-pseudorandom* if $\mathcal{D}$ contains a gap $\mathcal{C}$-pseudorandom language.

We remark that, if $\mathcal{C}$ is closed under complementation, our notion of gap $\mathcal{C}$-pseudorandomness is equivalent to that of $\mathcal{C}$-pseudorandomness. However, we cannot assume such a closure property for $\mathcal{C} = \mathrm{CFL}/n$.

[†]A language L is $\mathcal{C}$-pseudorandom if the function $\ell(n) = \left|\frac{dense(L\triangle A)(n)}{|\Sigma^n|} - \frac{1}{2}\right|$ is negligible [9].

An observation of a proof in [9], which establishes a close bridge between $\mathcal{C}$-pseudorandom languages and pseudorandom generators against $\mathcal{C}$, draws the fact that the proof actually shows, without any closure property, an equivalence between the gap $\mathcal{C}$-pseudorandomness and the existence of a pseudorandom generator against $\mathcal{C}$. We state this fact as a lemma below. For any function from Σ^* to Σ^*, its *range* $rang(G)$ is the set $\{G(w) \mid w \in \Sigma^*\}$.

Lemma 4.2 [9] *Let $\Sigma = \{0,1\}$. Let $\mathcal{C}$ be any language family. Let G be any function from Σ^* to Σ^* with the stretch factor $n+1$. Assume that G is almost 1-1. The function G is a pseudorandom generator against $\mathcal{C}$ if and only if the set $rang(G)$ is gap $\mathcal{C}$-pseudorandom.*

The above lemma roughly says that, as far as G is almost 1-1, the pseudorandomness of the generator G can be proven by establishing the gap pseudorandomness of the range of G. Using this lemma, we can give our proof of Proposition 3.3.

Proof of Proposition 3.3. Let G be any almost 1-1 pseudorandom generator against CFL. Now, assume that G belongs to $\mathrm{CFLSV_t}$. Let N be any npda with a write-only output tape computing G. By Lemma 4.2, the set $S =_{def} rang(G)$ is gap CFL-pseudorandom, implying that $S \notin \mathrm{CFL}$. We define a new npda M as follows. Let y be any input of length $n \geq 1$. We simulate N using imaginary input and output tapes as follows. When N reads a new bit written on its imaginary input tape, M guesses such a bit (either 0 or 1) and simulates N's step. As far as N's head keeps scanning the same tape cell, M uses the same bit without guessing another bit. If N writes down a bit b on its imaginary output tape, M first checks whether b appears on a cell where its head scans on its actual input tape, and then M simulates N's move. If b does not match the bit written on M's input tape, then M immediately rejects the input y; otherwise, M continues its simulation of N step by step. When N halts in an accepting state, then M also accepts the input.

If $y \in S$, then there exists a string x such that N on this input x produces y on its output tape along a certain accepting computation path, say, p. Consider an M's computation path in which M correctly guesses x and simulates N on the path p. Along this path, M obviously accepts y. On the contrary, when $y \notin S$, there is no string x or accepting path so that N on the input x correctly produces y. This means that M never accepts y in any computation path.

Therefore, M recognizes S. Since M is an npda, this draws a conclusion that S is context-free, contradicting our assumption that S is gap CFL-pseudorandom. $\Box$

Toward the end of this section, we shall prepare a useful language and its properties for the proof of our main theorem. We start with defining this language $IP_*^{(3)}$, which is founded on the *(binary) inner product*. The (binary) inner product between two binary strings x and y of length n is defined as $x \odot y = \sum_{i=1}^{n} x_i y_i$ if $x = x_1 x_2 \cdots x_n$ and $y = y_1 y_2 \cdots y_n$. The language $IP_*^{(3)}$ is then described as

$$IP_*^{(3)} = \{axyz \mid a \in \Sigma^{\leq 3}, x, y, z \in \Sigma^*, |x| = |z|, |y| = 2|x|, zx \odot y \equiv 1 \pmod 2\}.$$

Note that, in the above definition of $IP_*^{(3)}$, we use the term "$zx \odot y$" instead of "$xz \odot y$" because, otherwise, our proof given in later sections may not work properly. Figure 1 illustrates an example of string $axyz$ for $IP_*^{(3)}$ when $a = \lambda$.

We quickly discuss the complexity of the language $IP_*^{(3)}$. Consider the following simple algorithm for $IP_*^{(3)}$. Let $w = axyz$ be any input, provided that $|x| = |z|$, $|y| = 2|x|$, and y is of the form $y = y_1 y_2$ with $|y_1| = |y_2|$. First, we determine the size $|a|$. Consider the case where $a = \lambda$. First, mark the boundaries among x, y, and z in the input xyz. Computer $x \odot y_2 \pmod 2$ and $z \odot y_1 \pmod 2$ sequentially. Output the sum of these two products modulo two. It is not difficult to check that this algorithm takes time $O(n^2)$. The other case where $a \neq \lambda$ is similar. Hence, we obtain the following. We use the notation 1-DTIME$(t(n))$ to denote a "language" version of 1-FTIME$(t(n))$; that is, 1-DTIME$(t(n)) = \{A \mid \chi_A \in \text{1-FTIME}(t(n))\}$.

Lemma 4.3 *The language $IP_*^{(3)}$ belongs to* 1-DTIME$(O(n^2))$.

One of the most important ingredients of the proof of our main theorem is the gap CFL/n-pseudorandomness of $IP_*^{(3)}$.

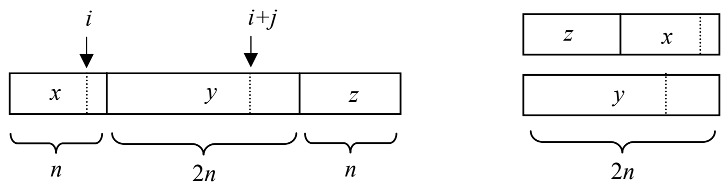

Figure 1: An input string xyz to $IP_*^{(3)}$, where i and j correspond to Lemma 6.1

Proposition 4.4 *The language $IP_*^{(3)}$ is gap* CFL/n*-pseudorandom.*

The proof of Proposition 4.4 will be given in Sections 6–8. An immediate corollary of Proposition 4.4 together with Lemma 4.3 is the gap CFL/n-pseudorandomness of the language family 1-DTIME($O(n^2)$). From this corollary, we also obtain a class separation: 1-DTIME($O(n^2)$) $\nsubseteq$ CFL/n, because $\mathcal{C}$ cannot be gap $\mathcal{C}$-pseudorandom if $\Sigma^* \in \mathcal{C}$.

Corollary 4.5 *The language family* 1-DTIME($O(n^2)$) *is gap* CFL/n*-pseudorandom.*

5 Proof of Theorem 3.2

Using Proposition 4.4, we shall prove Theorem 3.2, which states the existence of a pseudorandom generator G in 1-FTIME($O(n^2)$) against CFL/n. Our construction of G is rather straightforward from the definition of $IP_*^{(3)}$ and, therefore, the pseudorandomness of G follows directly from the gap pseudorandomness of $IP_*^{(3)}$.

First, we define a desired pseudorandom generator. Our input is of the form $w = axybz$ with $a \in \Sigma^{\leq 3}$, $b \in \Sigma$, $|x| = |bz|$, and $|y| = 2|x|$. Consider the simplest case where $a = \lambda$ and $|x| = n$. Define G as follows.

1 If $w = xybz$ and $zx \odot y \equiv 1 \pmod 2$, then let $G(w) = x\overline{b}ybz$.

2 If $w = xy1z$ and $zx \odot y \equiv 0 \pmod 2$, then let $G(w) = x1y1z$.

3 If $w = xy0z$ and $zx \odot y \equiv 0 \pmod 2$, then let i be the minimal index such that $y_i = 1$ if any.

 3a If i exists and $i \leq n-1$, then let $G(w) = x0y0\tilde{z}$, where $\tilde{z} = z_1z_2\cdots z_{i-1}\overline{z_i}z_{i+1}\cdots z_n$ if $z = z_1z_2\cdots z_i\cdots z_n$.

 3b If i exists and $n \leq i \leq 2n$, then let $G(w) = \tilde{x}0y0z$, where $\tilde{x} = x_1x_2\cdots x_{i-n-1}\overline{x_{i-n}}x_{i-n+1}\cdots x_n$ if $x = x_1x_2\cdots x_{i-n}\cdots x_n$.

 3c If i does not exist, then let $G(w) = x1y1z$.

If $a \neq \lambda$, then we define $G(w) = aG(xyz)$.

We need to prove that G satisfies the conditions necessary for a pseudorandom generator against CFL/n. Now, let us claim the following three properties.

Claim 1 *1. G is an almost 1-1 function.*

2. G is in 1-FTIME($O(n^2)$).

3. $rang(G) = IP_*^{(3)}$.

Proof. (1) By observing the definition of G, in all cases except for Case 3c, G is one-to-one on its domain. For each choice (x, z), however, G maps $\{x0^n1z, x0^n0z\}$ to $x10^n1z$, making itself two-to-one. Since the number of such pairs (x, y) is exactly 2^{2n}, we conclude that $|\{G(w) \mid w \in \Sigma^n\}| \geq 2^{4n} - 2^{2n} = 2^{4n}(1 - 1/2^{2n})$. Hence, G is almost 1-1.

(2) The proof of this claim is similar to Lemma 4.3.

(3) ($rang(G) \subseteq IP_*^{(3)}$) Let $u \in rang(G)$ and assume that $G(w) = u$ for a certain w. First, consider Case 2 of the definition of G. It thus follows that $u = x1y1z$, $zx \odot y \equiv 0 \pmod 2$, and $w = xy1z$. For convenience, we write $z' = 1z$ and $y' = 1y$. Since

$$z'x \odot y' = 1 \odot 1 + zx \odot y \equiv 1 + 0 = 1 \pmod 2,$$

we obtain $x'y'z \in IP_*^{(3)}$. Thus, w belongs to $IP_*^{(3)}$.

Next, consider Case 3b. In this case, we have $w = xy0z$, $zx \odot y \equiv 0 \pmod 2$, and $u = \tilde{x}0y0z$, where $\tilde{x} = x_1x_2\cdots x_{i-n-1}\overline{x_{i-n}}x_{i-n+1}\cdots x_n$ if $x = x_1x_2\cdots x_{i-n}\cdots x_n$ and $y_i = 1$ for the minimal index i. Let $y' = 0y$ and $z' = 0z$. For convenience, write $x^{(j)}$ to denote the x whose jth bit is removed. Similarly, we define $y^{(j)}$ from y. Note that

$$\begin{aligned} z'\tilde{x} \odot y' &= 0 \odot 0 + zx^{(i-n)} \odot y^{(i)} + \overline{x_{i-n}} \odot y_i \equiv zx^{(i-n)} \odot y^{(i)} + x_{i-n} \odot y_i + 1 \\ &= zx \odot y + 1 \equiv 0 + 1 = 1 \pmod 2 \end{aligned}$$

since $\overline{x_{i-n}} \odot y_i \equiv x_{i-n} \odot y_i + 1 \pmod 2$.

The other cases are similarly proven. Therefore, it holds that $rang(G) \subseteq IP_*^{(3)}$.

($rang(G) \supseteq IP_*^{(3)}$) Let $u \in IP_*^{(3)}$ and assume that $u = xy'z'$ with $z'x \odot y' \equiv 1 \pmod 2$. If $y' = by$ and $z' = \overline{b}z$ for a certain bit b, then it follows that $zx \odot y \equiv z'x \odot y' \equiv 1 \pmod 2$. Since this case corresponds to Case 1 of the definition of G, if we set $w = xybz$, then we obtain $G(w) = u$, indicating that $u \in rang(G)$.

Next, assume that $u = x'0y0z'$. Let $y = y_1y_2\cdots y_{2n}$. Now, consider the case where there exists the minimal index i such that $y_i = 1$. If $i \leq n-1$, then let $w = xy0z$, where $x = x'$ and z is such that $\tilde{z} = z'$. Let $z = z_1z_2\cdots z_n$. Write $z^{(i)}$ (resp., $y^{(i)}$) for the string obtained from z (resp., y) by flipping the ith bit. Clearly, it holds that

$$zx \odot y + 1 = z^{(i)}x \odot y^{(i)} + z_i \odot y_i + 1 \equiv z^{(i)}x \odot y + \overline{z_i} \odot y_i = z'x \odot y \equiv 1 \pmod 2,$$

from which we conclude that $zx \odot y \equiv 0 \pmod 2$. Since this is Case 3a, we obtain $G(w) = u$ and thus $u \in rang(G)$.

Since the other cases are similar, we have $IP_*^{(3)} \subseteq rang(G)$, as requested. □

Finally, we wish to show that G is indeed the desired pseudorandom generator. By Claim 1(1-2), G is an almost 1-1 function in 1-FTIME($O(n^2)$). To show that G fools every language in CFL/n, by Lemma 4.2 and Claim 1(3), it suffices to prove that $IP_*^{(3)}$ is gap CFL/n-pseudorandom. This gap pseudorandomness is given in Proposition 4.4. Therefore, the theorem holds.

6 Swapping Property of Context-Free Languages

We begin our proof of Proposition 4.4. This proof requires a unique property of context-free languages, which we call a *swapping property*. In this section, we shall show this useful property. Here, the notation Σ is used to denote an arbitrary alphabet.

Lemma 6.1 [swapping property lemma] *Let L be any context-free language over an arbitrary alphabet Σ with $|\Sigma| \geq 2$. For any triplet (j_0, k, n) of numbers with $j_0 \geq 2$ and $2j_0 \leq k \leq n$, there exist an alphabet Γ and two series $\{A_e\}_{e\in\Delta_{j_0,k,n}}$ and $\{B_e\}_{e\in\Delta_{j_0,k,n}}$, where $\Delta_{j_0,k,n} = \{i \in [0,n]_{\mathbb{Z}}, j \in [j_0,k]_{\mathbb{Z}}, i+j \leq n, u, v \in \Gamma\}$, which satisfy the following three conditions.*

(1) For any index tuple $(i,j,u,v) \in \Delta_{j_0,k,n}$, we have $A_{i,j,u,v} \subseteq \Sigma^{n-i-j} \times \Sigma^i$ and $B_{i,j,u,v} \subseteq \Sigma^j$.

(2) For every w with $|w| \geq 4$, $w \in L$ iff there exist an index tuple $(i,j,u,v) \in \Delta_{j_0,k,n}$ and three strings $x, y, z \in \Sigma^*$ such that $|x| = i$, $|y| = j$, $(z,x) \in A_{i,j,u,v}$, $y \in B_{i,j,u,v}$, and $w = xyz$.

(3) (swapping property) For every index $e \in \Delta_{j_0,k,n}$, if $(z_1,x_1,y_1),(z_2,x_2,y_2) \in A_e \times B_e$, then $(z_1,x_1,y_2),(z_2,x_2,y_1) \in A_e \times B_e$.

The reader may think it possible to obtain a similar result by following an argument used to prove the pumping lemma for context-free languages (see, e.g., [6] for the proof); however, such an argument may set j_0 to be a constant, which is independent of length n. For our proof of Proposition 4.4, we need j_0 to be chosen almost arbitrarily.

The following proof of Lemma 6.1 uses a certain characteristic feature of npda's.

Proof of Lemma 6.1. For the proof of the lemma, we need a specific simple form of npda's. Since $n \geq 4$, it is harmless to assume that L contains no empty string λ. Let us consider any npda $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$,

where Q is a set of internal states, Γ is a stack alphabet, δ is a transition function, $q_0 \in Q$ is the initial state, $z \in \Gamma$ is the stack start symbol, and $F \subseteq Q$ is a set of final states. For the functionality of these sets, see, e.g., [6]. For simplicity, we include two special *end-markers* ¢ and \$, which mark the left end and the right end of an input, respectively. Hereafter, we consider only inputs of the form $¢x\$$ with $x \in \Sigma^*$ and here we treat the endmarkers as a part of the input string x. For convenience, every tape cell is indexed with integers from left to right, and the left endmarker ¢ is always written in the 0th cell. Any input string x of length n is written in the cells indexed between 1 and n and the right endmarker \$ is written in the $n+1$st cell. Notice that $|¢x\$| = n+2$.

In what follows, we write the content of the stack of M as $s = s_1 s_2 s_3 \cdots s_m$ when the leftmost symbol s_1 is located at the top of the stack and the rightmost symbol s_m is at the bottom of the stack. For any string $x \in S$, $ACC(x)$ denotes the set of all accepting computation paths of M on the input x. In addition, we set $ACC_n = \bigcup_{x \in S} ACC(x)$.

Without loss of generality, we can impose the following restrictions on the behaviors of M. For the proof, see [5] (or [8]). We set $Q = \{q_0, q_1, q_f\}$, $z, S \in \Gamma$, $F = \{q_f\}$, and δ to satisfy the following four conditions.

1. $\delta(q_0, ¢, z) = \{(q_1, Sz)\}$.
2. $\delta(q_1, \$, z) = \{(q_f, z)\}$.
3. For any symbol $a \in \Sigma$, $\delta(q_1, a, z) = \emptyset$.
4. For any $a \in \Sigma$, any $v \in \Gamma$, and any $w \in \Gamma^*$, if $(q_1, w) \in \delta(q_1, a, v)$, then $|w| \leq 2$.

Furthermore, we introduce a few new terminologies. An *intercell boundary* i is a boundary or a border between two adjacent cells, the ith cell and the $i+1$st cell, in our npda's input tape. Fix a set $S \subseteq L \cap \Sigma^n$, a string x in S, and a computation path p in $ACC(x)$. Along such a path p, we assign to intercell boundary i a stack content produced after scanning the ith cell and before scanning the $i+1$st cell.

Now, we fix an arbitrary pair (j_0, k) satisfying that $0 \leq 2j_0 \leq k \leq n$. Recall the index set $\Delta_{j_0,k,n} = \{(i,j,u,v) \mid i \in [0,n]_{\mathbb{Z}}, j \in [j_0,k]_{\mathbb{Z}}, i+j \leq n, u,v \in \Gamma\}$ given in the lemma. Note that $|\Delta_{j_0,k,n}| \leq (n+1)^2|\Gamma|^2$. Now, we claim the following statement regarding M and $\Delta_{j_0,k,n}$.

Claim 2 *For every string $w \in L \cap \Sigma^n$, there exist an element $(i,j,u,v) \in \Delta_{j_0,k,n}$, a string s, and a path $p \in ACC(w)$ such that (1) $w = xyz$ with $|x| = i$ and $|y| = j$ and (ii) along p, M has a stack content us just after reading x and a stack content vs just after reading y, and s is never accessed by M during reading y. We call this s the* rooted stack content.

Assuming Claim 2, we continue our proof of Lemma 6.1. Let us define the desired series $\{A_e\}_{e \in \Delta_{j_0,k,n}}$ and $\{B_e\}_{e \in \Delta_{j_0,k,n}}$. For each index tuple $(i,j,u,v) \in \Delta_{j_0,k,n}$, we first define two sets $T^{(1)}_{i,u}$ and $T^{(2)}_{i,j,v}$ as follows.

- Let $T^{(1)}_{i,u}$ be the collection of pairs (x,s) with $s \in \Gamma^*$ and $|x| = i$ such that there exists a path $p \in ACC_n$ along which M produces a stack containing us just after reading $¢x$.
- Let $T^{(2)}_{i,j,v}$ be the collection of pairs (z,s) with $s \in \Gamma^*$ and $|z| = n-i-j$ such that M starts in inner state q_1 with a stack content vs and M enters an accepting or a final state q_f just after reading $z\$$.

The desired set $A_{i,j,u,v}$ is defined as $A_{i,j,u,v} = \{(z,x) \mid \exists s \in \Sigma^*[(x,s) \in T^{(1)}_{i,u} \ \wedge \ (z,s) \in T^{(2)}_{i,j,v}]\}$. Moreover, the set $B_{i,j,u,v}$ is defined as the collection of $y \in \Sigma^j$ such that there exist a stack content $s \in \Gamma^*$ and a path $p \in ACC_n$ along which M starts in state q_1 with a stack content us and produces a stack content vs after reading y, provided that M cannot access any symbol in s while reading y.

Clearly, we have $A_{i,j,u,v} \subseteq \Sigma^{n-i-j} \times \Sigma^j$ and $B_{i,j,u,v} \subseteq \Sigma^j$ and thus Condition 1 of the lemma holds. Condition 2 follows directly from Claim 2. Now, let us observe the following: for any two tuples $(z_1,x_1,y_1),(z_2,x_2,y_2) \in A_{i,j,u,v} \times B_{i,j,u,v}$, along certain accepting paths, M's behaviors during reading y_1 and y_2 are identical, except for their corresponding rooted stack contents. This makes it possible to swap y_1 and y_2 on these two paths without changing the outcomes of M. Therefore, it holds that $(z_1,x_1,y_2),(z_2,x_2,y_1) \in A_{i,j,u,v} \times B_{i,j,u,v}$, implying Condition 3.

Proof of Claim 2. We first review a result in [8]. Note that any accepting computation path of the npda M generates a length-$(n+2)$ series $(s_{-1}, s_0, s_1, \ldots, s_n, s_{n+1})$ of stack contents with $s_{-1} = s_n = s_{n+1} = z$ and $s_0 = Sz$. For any subinterval $I = [i_0, i_1]_{\mathbb{Z}}$ of I_0, we call a subsequence $\gamma = (s_{i_0}, s_{i_0+1}, \ldots, s_{i_1})$ a *stack*

transition (associated) with the interval I. The *height* at intercell boundary b of γ is the length $|s_b|$ of the stack content s_b at b. An *ideal* stack transition γ with an interval $[i_0, i_1]_{\mathbb{Z}}$ should satisfy that (i) we have the same height ℓ at both of the intercell boundaries i_0 and i_1 and (ii) all heights within this interval are more than or equal to ℓ.

Let $I = [i_0, i_1]_{\mathbb{Z}}$ be any subinterval of I_0 and let $\gamma = (s_{i_0}, s_{i_0+1}, \ldots, s_{i_1})$ be any ideal stack transition with this interval I. For each possible height ℓ, the *minimal width*, denoted $minwid_I(\ell)$ (resp., the *maximal width*, denoted $maxwid_I(\ell)$), is the minimal value (resp. maximal value) $|I'|$ (i.e., $|I'| = i'_1 - i'_0$) for which (i) $I' = [i'_0, i'_1]_{\mathbb{Z}} \subseteq I$, (ii) γ has height ℓ at both of the intercell boundaries i'_0 and i'_1, and (iii) at no intercell boundary $i \in I'$, γ has height less than ℓ.

The following lemma in [8] holds for any accepting computation path p of M.

Lemma 6.2 [8] *Let M be any npda that satisfies the conditions of this section. Let x be any string of length n accepted by M and let p be any computation path in $ACC(x)$. Assume that $j_0 \geq 2$ and $2j_0 \leq k \leq n$. Along the path p, for any interval $I = [i_0, i_1]_{\mathbb{Z}} \subseteq [-1, n+1]_{\mathbb{Z}}$ with $|I| > k$ and for any ideal stack transition γ with the interval I having height ℓ_0 at the two intercell boundaries i_0 and i_1, there are a subinterval $I' = [i'_0, i'_1]_{\mathbb{Z}}$ of I and a height $\ell \in [1, n]_{\mathbb{Z}}$ such that γ has height ℓ at both intercell boundaries i'_0 and i'_1, $j_0 \leq |I'| \leq k$, and $minwid_I(\ell) \leq |I'| \leq maxwid_I(\ell)$.*

Now, we prove the target claim. Let w be any string of length n accepted by M. We choose $i_0 = 0$ and $i_1 = n + 1$ and consider the interval $I = [i_0, i_1]_{\mathbb{Z}}$. Choose any ideal transition γ made by M along a certain accepting path in $ACC(w)$. Apply Lemma 6.2 and we take a subinterval $I' = [i'_0, i'_1]_{\mathbb{Z}}$ and a height $\ell \in [1, n]_{\mathbb{Z}}$ such that γ has height ℓ at both intercell boundaries i'_0 and i'_1, $j_0 \leq |I'| \leq k$, and $minwid_I(\ell) \leq |I'| \leq maxwid_I(\ell)$. Define $i = i'_0$ and $j = |I'|$. We decompose w as $w = xyz$ with $|x| = i$ and $|y| = j$. Let us assume that γ has a stack content us of length ℓ at the intercell boundary i'_0 (i.e., just after reading x) and similarly a stack content vs' of length ℓ at i'_1 (i.e., just after reading y). Since $minwid_I(\ell) \leq |I'| \leq maxwid_I(\ell)$, γ never has height less than ℓ between i'_0 and i'_1; that is, M cannot access any symbol in s. This implies that s' equals s. Therefore, the claim holds. □

This complete the proof of Lemma 6.1. □

7 Discrepancy Upper Bounds

Lemma 6.1 provides two useful series $\{A_e\}_e$ and $\{B_e\}_e$ for our proof of Proposition 4.4. In this section, we shall discuss a useful property concerning these series. For the sake of generality, we intend to write A and B respectively for arbitrary sets A_e and B_e.

To prove Proposition 4.4, we want to use a well-known discrepancy upper bound of the *inner-product-modulo-two* function. Through this section, for technicality, we deal only with strings of length $2n$ instead of length n. Let $x = x_1x_2 \cdots x_n$ and $y = y_1y_2 \cdots y_n$ be any two $2n$-bit strings. The *(binary) inner product* is a function defined as $x \odot y = \sum_{i=1}^{n} x_i y_i$.

Let M be a square matrix such that all entries are indexed by $\Sigma^{2n} \times \Sigma^{2n}$, where $\Sigma = \{0, 1\}$, and each (x, y)-entry has a value $x \odot y \pmod 2$. For convenience, we switch our values $\{0, 1\}$ to $\{1, -1\}$ and define the inner-product-modulo-two function f as $f(x, y) = (-1)^{x \odot y}$. Now, we introduce the notion of *discrepancy* over the matrix M.

Definition 7.1 For any set $T \subseteq \Sigma^{2n} \times \Sigma^{2n}$, the *discrepancy* of T is $Disc_M(T) = 2^{-4n} \left| \sum_{(x,y) \in T} f(x, y) \right|$.

In this paper, we need the following well-known upper bound of $Disc_M(T)$. How to obtain this bound is demonstrated in, e.g., [1].

Lemma 7.2 *For any two sets $A', B' \subseteq \Sigma^{2n}$, $Disc_M(A' \times B') \leq 2^{-3n} \sqrt{|A'||B'|}$.*

In our proof of Proposition 4.4, we need to consider five individual cases, among which two cases are essential to the proof. Each of those cases requires the discrepancy of a certain set T. Prior to the actual proof in Section 8, we shall examine those two major cases and give their associated discrepancy upper bounds, which are derived by an dexterous application of Lemma 7.2.

We begin with the following special case.

Case 1: Assume that (j, A, B) satisfy the following: $n \le j \le 2n$, $A \subseteq \Sigma^{4n-j}$ and $B \subseteq \Sigma^j$.

We define a set $T^{(j)}_{A,B}$ as follows: let $T^{(j)}_{A,B}$ be the collection of all pairs $(x, y) \in \Sigma^{2n} \times \Sigma^{2n}$ such that there exist six strings $x_1, x_2, x_3, y_1, y_2, y_3 \in \Sigma^*$ with $x = x_1x_2x_3$ and $y = y_1y_2y_3$ satisfying that $|x_1| = |y_1|$, $|x_2| = |y_2| = 2n - j$, $|x_3| = |y_3|$, $|x_1x_3| = j$, $y_2y_3x_1x_2 \in A$, and $x_3y_1 \in B$. Our key lemma is stated as follows.

Lemma 7.3 *For any fixed triplet (j, A, B) with $n \le j \le 2n$, $A \subseteq \Sigma^{4n-j}$, and $B \subseteq \Sigma^j$, we have $Disc_M\left(T^{(j)}_{A,B}\right) \le 2^{n-j}$ for any sufficiently large number $n \in \mathbb{N}$.*

Proof. Fix (j, A, B), as given in the lemma. Let $\ell = Disc_M(T^{(j)}_{A,B})$. Take any element $(x, y) \in T^{(j)}_{A,B}$ and consider their decompositions $x = x_1x_2x_3$ and $y = y_1y_2y_3$, as stated above. Since we cannot apply Lemma 7.2 directly to $T^{(j)}_{A,B}$, we need to find a different way to view $T^{(j)}_{A,B}$.

First, we introduce an index set $D_{j,n} = \{(x_2, y_2) \mid x_2, y_2 \in \Sigma^{2n-j}\}$, which clearly satisfies $|D_{j,n}| = 2^{2(2n-j)}$. Fixing each index pair (a, b) in $D_{j,n}$, we further introduce two new sets $A_{a,b}$ and $B_{a,b}$ as follows.

- $A_{a,b} = \{x_1ay_3 \mid \exists x_3, y_1 \in \Sigma^* \text{ s.t. } (x_1ax_3, y_1by_3) \in T^{(j)}_{A,B}\}$.
- $B_{a,b} = \{y_1bx_3 \mid \exists x_1, y_3 \in \Sigma^* \text{ s.t. } (x_1ax_3, y_1by_3) \in T^{(j)}_{A,B}\}$.

Notice that $|A_{a,b}| \le 2^j$ since $|a| = 2n - j$. Similarly, we have $|B_{a,b}| \le 2^j$. Next, we claim that $T^{(j)}_{A,B}$ is "fundamentally" identical to the union $\bigcup_{(a,b)\in D_{j,n}}(A_{a,b} \times B_{a,b})$ over the matrix M. This is stated more precisely in the following claim.

Claim 3 *There is a one-to-one map μ from $T^{(j)}_{A,B}$ to $\bigcup_{(a,b)\in D_{j,n}}(A_{a,b} \times B_{a,b})$ such that, for any pair $(x, y) \in T^{(j)}_{A,B}$, if $\mu(x, y) = (x', y')$ then $f(x, y) = f(x', y')$.*

Proof. Let (x, y) be any pair in $T^{(j)}_{A,B}$ with $x = x_1x_2x_3$ and $y = y_1y_2y_3$, as before. We then define $\mu(x, y) = (x', y')$, where $x' = x_1x_2y_3$ and $y' = y_1y_2x_3$. Obviously, (x', y') belongs to $A_{x_2,y_2} \times B_{x_2,y_2}$. The one-oneness of μ is trivial from the definition of μ. Moreover, since

$$x \odot y = x_1x_2 \odot y_1y_2 + x_3 \odot y_3 = x_1x_2y_3 \odot y_1y_2x_3 = x' \odot y',$$

it follows that $f(x, y) = f(x', y')$. □

The above one-to-one map μ helps us estimate the value ℓ as

$$\begin{aligned}
\ell &= Disc_M\left(T^{(j)}_{A,B}\right) = 2^{-4n}\left|\sum_{(x,y)\in T^{(j)}_{A,B}} f(x,y)\right| \\
&\le 2^{-4n}\sum_{(a,b)\in D_{j,n}}\left|\sum_{(x',y')\in A_{a,b}\times B_{a,b}} f(x',y')\right| \le \sum_{(a,b)\in D_{j,n}} Disc_M(A_{a,b}\times B_{a,b}),
\end{aligned}$$

which is, by Lemma 7.2, further bounded by

$$\ell \le 2^{-3n}\sum_{(a,b)\in D_{j,n}}\sqrt{|A_{a,b}||B_{a,b}|} \le 2^{-3n}|D_{j,n}|\max_{(a,b)\in D_{j,n}}\left\{\sqrt{|A_{a,b}||B_{a,b}|}\right\} \le 2^{-3n}2^{4n-2j}2^j = 2^{n-j}.$$

□

Case 2: Assume that (j, A, B) satisfies the following: $2n < j \le 3n$, $A \subseteq \Sigma^{4n-j}$, and $B \subseteq \Sigma^j$.

Different from Case 1, we define $T^{(j)}_{A,B}$ as the set of all pairs $(x, y) \in \Sigma^{2n} \times \Sigma^{2n}$ with $x = x_1x_2x_3$ and $y = y_1y_2y_3$ satisfying that $|x_1| = |y_1|$, $|x_2| = |y_2| = 4n - j$, $|x_3| = |y_3|$, $|x_1y_1y_2y_3x_3| = j$, $x_2 \in A$, and $x_3y_1y_2y_3x_1 \in B$. We want to show the following claim.

Lemma 7.4 *For any fixed triplet (j, A, B) with $2n < j \leq 3n$, $A \subseteq \Sigma^{4n-j}$, and $B \subseteq \Sigma^j$, we have $Disc_M\left(T^{(j)}_{A,B}\right) \leq 2^{j-3n}$ for any sufficiently large number $n \in \mathbb{N}$.*

Proof. Our index set is defined as $E_{j,n} = \{(x_1, y_1, x_3, y_3) \mid |x_1| = |y_1|, |x_3| = |y_3|, |x_1x_3| = j - 2n\}$. Notice that $|E_{j,n}| = 2^{2(j-2n)}$. Now, for each index $e = (a, b, c, d) \in E_{j,n}$, we define two sets A_e and B_e as follows.

- $A_e = \{ax_2c \mid \exists y_2 \in \Sigma^* \text{ s.t. } (ax_2c, by_2d) \in T^{(j)}_{A,B}\}$.
- $B_e = \{by_2d \mid \exists x_2 \in \Sigma^* \text{ s.t. } (ax_2c, by_2d) \in T^{(j)}_{A,B}\}$.

Since $|ac| = |bd| = j - 2n$, it follows that $|A_e| \leq 2^{4n-j}$ and $|B_e| \leq 2^{4n-j}$.

Moreover, from the definition of A_e and B_e, we have the following property: if $(x, y) \in T^{(j)}_{A,B}$ with $x = x_1x_2x_3$ and $y = y_1y_2y_3$, then $(x, y) \in A_{(x_1,y_1,x_3,y_3)} \times B_{(x_1,y_1,x_3,y_3)}$. In other words, $T^{(j)}_{A,B} \subseteq \bigcup_{e \in E_{j,n}} A_e \times B_e$. From this property, we have

$$\ell = Disc_M\left(T^{(j)}_{A,B}\right) \leq \sum_{e \in E_{j,n}} Disc_M(A_e \times B_e),$$

which is bounded by

$$\ell \leq 2^{-3n} \sum_{e \in E_{j,n}} \sqrt{|A_e||B_e|} \leq 2^{-3n}|E_{j,n}| \max_{e \in E_{j,n}} \left\{\sqrt{|A_e||B_e|}\right\} \leq 2^{-3n}2^{2j-4n}2^{4n-j} = 2^{j-3n}.$$

This completes the proof. □

8 Proof of Proposition 4.4

Finally, we shall give the proof of Proposition 4.4, which states that $IP^{(3)}_*$ is gap CFL/n-pseudorandom.

Let S be any language over $\Sigma = \{0, 1\}$ in CFL/n and let p be any non-zero polynomial. Choose any sufficiently large number n so that the following argument holds for it. Consider the major case where $|axyz| = 4n$ with $a = \lambda$. For notational convenience, let $U_1 = S \cap IP^{(3)}_* \cap \Sigma^{4n}$ and $U_0 = S \cap \overline{IP^{(3)}_*} \cap \Sigma^{4n}$. Since

$$dense(IP^{(3)}_* \cap S)(4n) - dense(\overline{IP^{(3)}_*} \cap S)(4n) = |U_1| - |U_0|,$$

our goal now is to verify that $||U_1| - |U_0|| \leq 2^{4n}/8p(n)$.

Since $S \in$ CFL/n, we take an advice function h and a language $S' \in$ CFL over another alphabet $\tilde{\Sigma}$ such that, for every string $x \in \Sigma^*$, $x \in S$ iff $\left[\begin{smallmatrix} x \\ h(|x|) \end{smallmatrix}\right] \in S'$. We define our magic numbers j_0 and k as $j_0 = 4n/3$ and $k = 2j_0$ $(= 8n/3)$, and apply Lemma 6.1 with these numbers. Unlike Section 6, here we use "$4n$," instead of "n," as the length of our input strings. To simplify our notation further, we write $\Delta_{j_0,k}$ for $\Delta_{j_0,k,4n}$ by dropping the subscript "$4n$." Note that $|\Delta_{j_0,k}| \leq (4n+1)^2|\Gamma|^2 < 2^{3\log n}$ since n is assumed to be sufficiently large. Moreover, let us take two series $\{A_e\}_{e \in \Delta_{j_0,k}}$ and $\{B_e\}_{e \in \Delta_{j_0,k}}$ given in Lemma 6.1 and fix an index $e = (i, j, u, v) \in \Delta_{j_0,k}$ for the following argument.

Since n is fixed, we want to argue how to remove the advice string $h(4n)$ from the rest of our proof. Assuming that $h(4n) = h_1h_2h_3$ with $|h_1| = i$ and $|h_2| = j$, we define two sets $A'_e = \{zx \mid \left[\begin{smallmatrix} zx \\ h_3h_1 \end{smallmatrix}\right] \in A_e\}$ and $B'_e = \{y \mid \left[\begin{smallmatrix} y \\ h_2 \end{smallmatrix}\right] \in B_e\}$. Unless there is any confusion, for our convenience, we denote these sets A'_e and B'_e simply by A_e and B_e, respectively. With this abbreviation, it holds that $A_e \subseteq \Sigma^{4n-i-j} \times \Sigma^j$ and $B_e \subseteq \Sigma^j$. By identifying a pair (z, x) with a string zx, we further treat A_e as a subset of Σ^{4n-j}; hereafter, we can assume that $A_e \subseteq \Sigma^{4n-j}$. Finally, let us introduce a set S_e to be the collection of all strings w in $S \cap \Sigma^{4n}$ satisfying that $z'x' \in A_e$ and $y' \in B_e$, where $x' = pref_i(w)$, $y' = midd_{i,i+j}(w)$, and $z' = suf_{4n-i-j}(w)$.

Claim 4 $||U_1| - |U_0|| \leq \sum_{e \in \Delta_{j_0,k}} ||U_1 \cap S_e| - |U_0 \cap S_e||$.

This claim is shown as follows. Lemma 6.1 leads to the equality $S \cap \Sigma^{4n} = \bigcup_{e\in\Delta_{j_0,k}} S_e$, from which we obtain

$$||U_1| - |U_0|| = \left| \sum_{e\in\Delta_{j_0,k}} |U_1 \cap S_e| - \sum_{e\in\Delta_{j_0,k}} |U_0 \cap S_e| \right| \leq \sum_{e\in\Delta_{j_0,k}} ||U_1 \cap S_e| - |U_0 \cap S_e|| .$$

Next, we want to show that, for every index $e \in \Delta_{j_0,k}$, $||U_1 \cap S_e| - |U_0 \cap S_e|| \leq 2^{11n/3}$. To show this inequality, we need to consider five separate cases, of which the most crucial cases are Cases 1 and 2 given below.

Case 1: Assume that $0 \leq i \leq n$ and $j_0 \leq j \leq 2n$.

In this special case, we limit our attention to the index set $\Delta^{(1)}_{j_0,k} = \{(i,j,u,v) \in \Delta_{j_0,k} \mid 0 \leq i \leq n, j_0 \leq j \leq 2n\}$. To estimate the value $m(e) = ||U_1 \cap S_e| - |U_0 \cap S_e||$ for each index $e \in \Delta^{(1)}_{j_0,k}$, we consider a set $T_e = T^{(j)}_{A_e,B_e}$, where $T^{(j)}_{A,B}$ is defined in Case 1 of Section 7. The following claim gives a direct translation between S_e and T_e.

Claim 5 *For any index $e \in \Delta^{(1)}_{j_0,k}$ and any three strings $x,y,z \in \Sigma^*$ with $|x| = |z| = n$ and $|y| = 2n$, $xyz \in S_e$ if and only if $(zx,y) \in T_e$.*

Proof. Let $e = (i,j,u,v) \in \Delta^{(1)}_{j_0,k}$ and let $w = xyz$ be any string satisfying that $|x| = |z| = n$ and $|y| = 2n$. Now, we partition w as $w = x'y'z'$ with three strings $x' = pref_i(w)$, $y' = midd_{i,i+j}(w)$, and $z' = suf_{4n-i-j}(w)$. Since $0 \leq i \leq n$ and $n < j \leq 2n$, we can express x and y as $x = x_1x_2$ and $y = y_1y_2$ that satisfy $x' = x_1$, $y' = x_2y_1$, and $z' = y_2z$.

(Only If–part) Assume that $w \in S_e$. This makes the pair $(z'x', y')$ belong to $A_e \times B_e$. Equivalently, (y_2zx_1, x_2y_1) is in $A_e \times B_e$. From the definition of $T^{(j)}_{A_e,B_e}$, it follows that $(zx_1x_2, y_1y_2) \in T^{(j)}_{A_e,B_e}$, which is obviously equal to $(zx,y) \in T_e$.

(If–part) Assume that $(zx,y) \in T_e$; in other words, (zx_1x_2, y_1y_2) is in $T^{(j)}_{A_e,B_e}$. This implies that $(y_2zx_1, x_2y_1) \in A_e \times B_e$, which is also equivalent to $(z'x', y') \in A_e \times B_e$. From this, we can conclude that $xyz \in S_e$. □

For the estimation of $m(e)$, we prove the following useful statements.

Claim 6 *1. For each index $e \in \Delta^{(1)}_{j_0,k}$, $||U_1 \cap S_e| - |U_0 \cap S_e|| = 2^{4n} Disc_M(T_e)$.*

2. For each index $e \in \Delta^{(1)}_{j_0,k}$, $Disc_M(T_e) \leq 2^{-n/3}$.

Proof. (1) Let e be any index in $\Delta^{(1)}_{j_0,k}$. Recall the inner-product-modulo-two function f. For the desired statement, we first note that $f(zx,y) = 1$ iff $xyz \in IP^{(3)}_*$. By a translation between S_e and T_e given in Claim 5, it immediately follows that, for any bit b,

$$|U_b \cap S_e| = |\{(zx,y) \in T_e \mid f(zx,y) = b\}| = |T_e \cap f^{-1}(b)|.$$

Using this equality, we calculate the value $2^{4n} Disc_M(T_e)$ as follows:

$$\begin{aligned} 2^{4n} Disc_M(T_e) &= \left| \sum_{(x,y)\in T_e} f(x,y) \right| = \left| \sum_{(x,y)\in T_e\cap f^{-1}(1)} 1 - \sum_{(x,y)\in T_e\cap f^{-1}(0)} 1 \right| \\ &= \left| |T_e \cap f^{-1}(1)| - |T_e \cap f^{-1}(0)| \right| = ||U_0 \cap S_e| - |U_1 \cap S_e||. \end{aligned}$$

(2) Let $e \in \Delta^{(1)}_{j_0,k}$. Since $T_e = T^{(j)}_{A_e,B_e}$, Lemma 7.3 implies that $Disc_M(T_e) \leq 2^{n-j}$. From our assumption $j \geq j_0$, it follows that $Disc_M(T_e) \leq 2^{5-j_0} = 2^{-n/3}$. □

From Claim 6 follows

$$m(e) = ||U_1 \cap S_e| - |U_0 \cap S_e|| = 2^{4n} Disc_M(T_e) \leq 2^{4n} 2^{-n/3} = 2^{11n/3}.$$

Case 2: Assume that $0 \le i \le n$ and $2n < j \le k$.

Let $\Delta^{(2)}_{j_0,k} = \{(i,j,u,v) \in \Delta_{j_0,k} \mid 0 \le i \le n, 2n < j \le k\}$. Again, for each index $e \in \Delta^{(2)}_{j_0,k}$, we consider $T_e = T^{(j)}_{A_e,B_e}$, where we defined $T^{(j)}_{A_e,B_e}$ in Case 2 of Section 7. Now, we want to show that $m(e) = ||U_1 \cap S_e| - |U_0 \cap S_e|| \le 2^{11n/3}$. We begin with the following claim, which is similar to Claim 5.

Claim 7 *For any $e \in \Delta^{(2)}_{j_0,k}$ and any triplet x,y,z with with $|x| = |z| = n$ and $|y| = 2n$, $xyz \in S_e$ iff $(zx, y) \in T_e$.*

Proof. Let $w = xyz$ be any string with $|x| = |z| = n$ and $|y| = 2n$. Let $x' = pref_i(w)$, $y' = midd_{i,i+j}(w)$, and $z' = suf_{4n-i-j}(w)$. Using our condition that $0 \le i \le n$ and $2n < j < 3n$, the strings x and z can be expressed as $x = x_1x_2$ and $z = z_1z_2$ with the conditions that $x' = x_1$, $y' = x_2yz_1$, and $z' = z_2$.

(Only If–part) Let us assume that $w \in S_e$. We then have $(z'x', y') \in A_e \times B_e$. This is equivalent to $(z_2x_1, x_2yz_1) \in A_e \times B_e$. This implies that $(z_1z_2x_1x_2, y) \in T^{(j)}_{A_e,B_e}$. Clearly, we obtain $(zx, y) \in T_e$.

(If–part) Assume that $(zx, y) \in T_e$. This means that $(z_2z_1z_2x_1x_2, y) \in T^{(j)}_{A_e,B_e}$. Hence, by the definition of $T^{(j)}_{A_e,B_e}$, we have $(z_2x_1, x_2yz_1) \in A_e \times B_e$. In other words, $(z'x', y') \in A_e \times B_e$, from which we obtain $xyz \in S_e$. $\square$

Note that, by Lemma 7.4, since $2n < j \le k$, we have

$$Disc_M(T_e) \le 2^{j-3n} \le 2^{k-3n} = 2^{-n/3}.$$

Thus, similar to Case 1, we obtain $m(e) = ||U_1 \cap S_e| - |U_0 \cap S_e|| \le 2^{11n/3}$, as requested.

Case 3: Assume that $n < i < 2n$ and $7n/3 < i + j \le 3n$.
This case is similar to Case 2.

Case 4: Assume that $n < i < 2n$ and $3n < i + j \le 4n$.
This case is similar to Case 1.

Case 5: Assume that $2n \le i \le 8n/3$ and $i + j \le 4n$.
This case is similar to Case 3.

Overall, we conclude that, for every index $e \in \Delta_{j_0,k}$, $m(e) = ||U_1 \cap S_e| - |U_0 \cap S_e|| \le 2^{11n/3}$. Recall that $|\Delta_{j_0,k}| < 2^{3\log n}$. It thus follows that

$$\begin{aligned} ||U_1| - |U_0|| &\le \sum_{e \in \Delta_{j_0,k,n}} ||U_1 \cap S_e| - |U_0 \cap S_e|| \le |\Delta_{j_0,k}| \cdot 2^{11n/3} \\ &\le 2^{3\log n} 2^{11n/3} < 2^{15n/4}, \end{aligned}$$

which is clearly bounded from above by $2^{4n}/8p(n)$.

Finally, we need to consider the remaining case where $a \neq \lambda$. Let $\ell = |a| \le 3$ and define a set $S_a = \{xyz \mid axyz \in S\}$. We write aS_a for the set $\{aw \mid w \in S_a\}$. It is easy to check that $dense(IP^{(3)}_* \cap aS_a)(4n + \ell) = dense(IP^{(3)}_* \cap S_a)(4n)$ and $dense(\overline{IP^{(3)}_*} \cap aS_a)(4n + \ell) = dense(\overline{IP^{(3)}_*} \cap S_a)(4n)$. Since $S = \bigcup_{a \in \Sigma^\ell} aS_a$, we thus obtain

$$\begin{aligned} &\left| dense(IP^{(3)}_* \cap S)(4n + \ell) - dense(\overline{IP^{(3)}_*} \cap S)(4n + \ell) \right| \\ &\quad\le \sum_{a \in \Sigma^\ell} \left| dense(IP^{(3)}_* \cap S_a)(4n) - dense(\overline{IP^{(3)}_*} \cap S_a)(4n) \right| \\ &\quad\le 2^\ell \cdot \frac{2^{4n}}{8p(n)} \le \frac{2^{4n}}{p(n)}. \end{aligned}$$

Since p is arbitrary, we then conclude that $IP^{(3)}_*$ is gap CFL/n-pseudorandom. This completes the proof of Proposition 4.4.

9 Conclusion

Pseudorandom generators have played an essential role in modern cryptography. Through this paper, we have discussed such generators in a framework of formal language theory. It was shown in [9] that a pseudorandom generator against REG/n exists and such a generator can be found in $\mathrm{CFLSV_t}$. As a next step toward a full understanding of pseudorandomness in formal language theory, our main contribution of this paper is to have given a proof of the existence of a certain type of pseudorandom generator against CFL/n within 1-FTIME($O(n^2)$) but not in $\mathrm{CFLSV_t}$.

To close this paper, we raise a fresh question of whether "easy-to-compute" pseudorandom generators exist against $\mathrm{CFL}(k)/n$, where $\mathrm{CFL}(k)$ consists of languages made by k intersections of context-free languages (see [9]).

References

[1] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[2] Y. Bar-Hillel, M. Perles, and E. Shamir. On formal properties of simple phrase-structure grammars. *Z. Phonetik Sprachwiss. Kommunikationsforsch*, 14, 143–172, 1961.

[3] M. Blum and S. Micali. How to genrate cryptographically strong sequences of pseudorandom bits. *SIAM J. on Comput.*, 13, 850–864, 1984.

[4] O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press. 2001.

[5] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, 1979.

[6] J. E. Hopcroft, R. Motwami, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation (2nd edition)*, Addison-Wesley, 2001.

[7] K. Tadaki, T. Yamakami, and J. C. H. Lin. Theory of one tape linear time Turing machines. In *Proceedings of the 30th SOFSEM Conference on Current Trends in Theory and Practice of Computer Science*, Lecture Notes in Computer Science, Vol.2932, pp.35–348, 2004. A detailed version is available on line at arXiv:cs/0310046.

[8] T. Yamakami. Swapping lemmas for regular and context-free languages. Available on line at arXiv:0808.4122, 2008.

[9] T. Yamakami. Immunity and pseudorandomness of context-free languages. Available on line at arXiv:0902.0261, 2009.

[10] A. C. Yao. Theory and application of trapdoor functions. *Proc. 23rd IEEE Symposium on Foundations of Computer Science*, pp.80–91, 1982.